Mobile Manipulation in Unstructured Environments: Perception, Planning, and Execution

Article *in* IEEE Robotics & Automation Magazine · June 2012

DOI: 10.1109/MRA.2012.2191995

4 authors, including:

Sachin Chitta
SRI International
76 PUBLICATIONS **5,686** CITATIONS

Perception, Planning, and Execution for Mobile Manipulation in Unstructured Environments

Sachin Chitta[1] E. Gil Jones[1] Matei Ciocarlie[1] Kaijen Hsiao[1]

***Abstract*—Unstructured human environments present a substantial challenge to effective robotic operation. Mobile manipulation in typical human environments requires dealing with novel unknown objects, cluttered workspaces, and noisy sensor data. We present an approach to mobile pick and place in such environments using a combination of 2D and 3D visual processing, tactile and proprioceptive sensor data, fast motion planning, reactive control and monitoring, and reactive grasping. We demonstrate our approach by using a two-arm mobile manipulation system to pick and place objects. Reactive components allow our system to account for uncertainty arising from noisy sensors, inaccurate perception (e.g. object detection or registration) or dynamic changes in the environment. We also present a set of tools that allow our system to be easily configured within a short time for a new robotic system.**

I. Introduction

Human environments provide incredible challenges for effective mobile manipulation. The remarkable dexterity and mobility of humans results in environments where tasks require a high degree of flexibility in perception, motion, and control that robots currently lack. Large variations in objects, lighting, and clutter make household manipulation a very difficult problem. Most households have very limited space for robots to move around in, and objects are often hidden in clutter, behind other objects, or inside containers. Furthermore, with humans moving around in the environment, safety is an important consideration for any household manipulation robot. Mobile manipulation in such environments will require a tightly-integrated effort, combining techniques in perception, motion planning, grasping, and control.

Significant progress has been made in recent years in advancing the state of the art in mobile manipulation. Mobile manipulation platforms such as the PR2 (see Fig. 1), Intel HERB Personal Robot, ICub, the TUM Rosie robot, the HRP2, ARMAR, and Justin have demonstrated a variety of integrated manipulation tasks. Newer systems such as the PR2 have integrated 3D sensors to build a consistent representation of their changing environments. 2D visual information has been used for object recognition and tracking, and as feedback for visual servoing controllers. However, visual information is often not sufficient for robust operation; robustness can be increased with the addition of tactile and proprioceptive feedback. Safe, collision-free motion can be achieved by combining fast, realtime motion planning with smooth, reactive controllers.

[1] Sachin Chitta, E. Gil Jones, Matei Ciocarlie and Kaijen Hsiao are with Willow Garage Inc., Menlo Park, CA 94025, USA (sachinc,gjones,matei,hsiao)@willowgarage.com

Fig. 1. The PR2 (right) and the Care-O-Bot (left). Both robots are executing a tabletop pick and place task using the approach described in this paper.

In this paper, we present an integrated approach to manipulation and grasping in household environments. We focus specifically on mobile pick and place tasks, i.e., tasks where a robot is required to move objects in human, unstructured environments. Our approach starts by building a consistent model of the environment using input from both 3D and 2D visual sensors. The environment is represented using a combination of pre-generated mesh models for known objects and a voxel-based representation for the other parts of the environment. The environment representation serves as input to motion planners that generate collision-free, smooth motions for the robot. Fast, realtime trajectory controllers are used to perform the planned trajectories on the robot. Grasp selection for objects consists either of selection of an appropriate grasp from a set of grasps pre-computed offline, or computation of grasps online.

The approach presented here builds on earlier work in [1], adding several new components. The primary contribution is an integrated framework for mobile pick and place, demonstrated on multiple robotic systems and built on a robust, reusable software framework that incorporates capabilities in navigation, perception and manipulation. A motion planner for dealing with geometric constraints for 7 degree of freedom (DOF) arms allows for executing constrained pick and place tasks. A new grasp planning framework developed here is capable of dealing with uncertainty, especially arising from noise in object recognition and registration. The ability to react to unforeseen obstacles in the environment based on realtime representation of the environment is another contribution. Finally, the development of tools that allow our framework to

be easily adapted to new robotic systems, and a set of standard interfaces where new approaches to capabilities like object recognition, motion planning and grasp planning can be easily plugged in is, we believe, one of the strongest contributions of our work.

II. Related Work

An exhaustive survey of mobile manipulation systems and techniques is beyond the scope of this paper; however, we will mention a few recent efforts. The Intel HERB mobile manipulation platform has demonstrated impressive capabilities in indoor environments ranging from being able to pick and place objects [2] to push-based manipulation on tabletop environments [3]. The DLR Justin robot has demonstrated exceptional manipulation capabilities, including the ability to catch balls [4] and make coffee. The ARMAR-III robots have been used for tasks in a prototype kitchen setting demonstrating impressive capabilities including combined grasp and motion planning [5]. Other examples of integrated mobile manipulation systems that have demonstrated autonomous behavior include El-E [6], STAIR [7], Care-O-Bot [8], the TUM Rosie robot [9], and the PR2 robot [10].

The current state of the art in mobile pick and place is the result of significant recent interest in the area. The work of Srinivasa, et al [11] is most similar to our approach, integrating motion planning and perception for mobile pick and place. Learning based approaches have found success in [7] where a robot was able to learn grasp points for several objects. Web-based instructions were used for a breakfast making task in [9] using the TUM Rosie robot. The ARMAR series of robots have also demonstrated impressive capabilities in a home scenario [5], using an integrated grasp and motion planning approach.

Our approach to mobile pick and place is distinguished from the state-of-the-art systems mentioned here by the number and variety of integrated components, as well as by the availability and reusability of our software framework. Perceptual input is fully integrated into our system and plays a major role in every component. Our system is reactive, using information from all available sensors, including visual and proprioceptive sensors, to react to changes in the environment or errors in perception. Our approach to grasping can deal with a variety of unknown objects, i.e. objects for which the robot has no prior information, as well as objects for which 3D models are available. Our approach is also available as a well-packaged, stable software framework, available for download and use on robots. The framework includes well defined interfaces between components, making it extensible and easy to plugin new approaches to grasping, motion planning and perception. The framework also includes tools that allow the crucial parts of our approach to be easily ported to multiple manipulation platforms. Section VI, in particular, describes a tool that makes this software framework truly robot agnostic. Finally, the effectiveness of our approach has been validated by its demonstration on multiple robotic platforms, including the PR2, the Care-O-Bot (as described in Section VI) and multiple industrial platforms [12], [13].

A. Structure of this paper

This paper is structured as follows: In Section III, we give an overview of our approach and the robot that we use for experiments in Section III. In Section IV, we present details on the motion planning, control, and execution components used in our approach. In Section V, we expand on our approach to grasping. In Section VI, we detail our efforts to make tools that allow for easy portability of our approach to different mobile manipulation platforms. Finally, in Section VII, we present experimental results and applications of our approach to different mobile manipulation tasks.

III. Overview

The primary motivating focus of this work is mobile pick and place, i.e., the ability to pick up objects and move them to new locations in an environment. Our goal is to develop an approach to this task that is suited for unstructured environments. Towards this end, our approach makes extensive use of information from the sensors of the robot to model the environment online. We also intend for our approach to be easily deployable on different mobile manipulation systems. We have developed a software framework (built on ROS [14], PCL (pointclouds.org), and OpenCV (opencv.willowgarage.com)) and a set of software tools that make porting our approach to other mobile manipulation systems much easier. Our framework is modular, with the intention of allowing incorporation of alternate components for motion planning, control, grasping, or perception.

The mobile manipulation platform that we use for validating our approach is the PR2 (Fig. 1), which has an omni-directional base and two 7-DOF arms. It is equipped with a tilting laser scanner mounted on the body, and two stereo cameras and a texture projector that are mounted on a pan-tilt platform (the head). An additional laser scanner mounted on the base and a body-mounted IMU are used extensively for localization and navigation. Encoders on each joint provide joint angle information. The end-effector is a parallel jaw gripper whose fingertips are equipped with capacitive sensor arrays, each consisting of 22 individual cells; the gripper also contains an accelerometer. For some of our work, we have also utilized an additional RGBD sensor (a Microsoft Kinect®) mounted on the head of the robot.

Our approach utilizes ROS for all communication and configuration needs. Generic interfaces are defined between different components in our framework as ROS interfaces. Any new component, e.g. a new grasp planning algorithm, can be easily incorporated into the system if it implements the same interface. For example, several groups have been able to integrate their custom motion planners within our framework for execution on the PR2 robot. This includes CHOMP [15], Search-based planners [16], Learning dimensional descent [17], and STOMP [18].

The overall approach to the pick and place task is shown schematically in Fig. 2. A representation of the environment is built and maintained using a combination of 2D and 3D visual sensors. This model includes a lower resolution representation of the environment using an octree-based approach, and higher

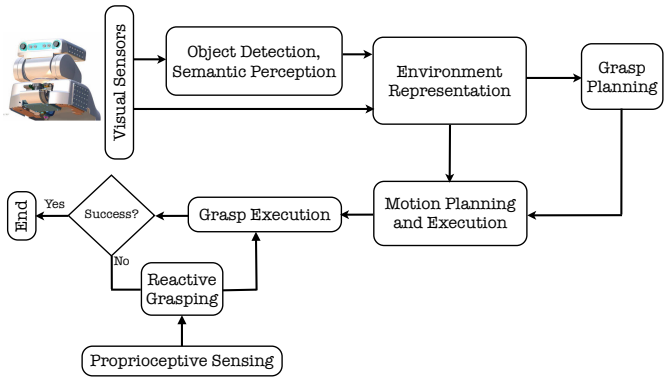

Fig. 2. High-level system architecture for our approach to executing pick and place tasks.

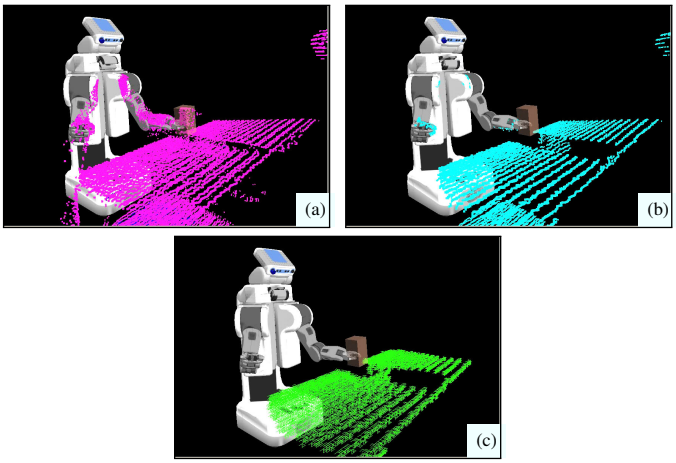

Fig. 3. Raw sensor data (a) is filtered to remove spurious points (b) and points associated with the robot body (c). Note that the object held in the hand is also considered a part of the robot body. (This figure is best viewed in color).

resolution representations of objects of interest using mesh primitives. Object detection and segmentation routines are used to segment the raw point cloud representation and to recognize objects of interest, including planar surface models from which objects can be placed or picked up. The environment model serves as input to both the grasp planning and motion planning components. The grasp planning framework determines grasps for objects of interest and is capable of dealing with either detailed mesh representations or a coarser model obtained from segmentation. The motion planning and execution framework plans and executes trajectories to the desired end-effector positions for grasping, pickup, and placement of objects. A grasp execution component takes over control of the arm during the actual grasping action, using reactive controllers to correct for any error in the perceived placement of the objects. These components will now be described in greater detail.

IV. Environment Modeling, Motion Planning and Execution

Our approach to motion planning and execution builds on online modeling of the environment, randomized motion planners, and the use of visual sensors for dealing with disturbances. Our focus is on developing a *reliable* framework for moving the robot while taking into account any obstacles that may be present in the environment. We also aim to account for possible disturbances during execution by either pausing and re-executing planned trajectories or replanning when necessary.

A. Environment Modeling

Building a consistent and detailed model of the environment is essential for any operation in an unstructured environment. The process starts with raw sensor data from 3D visual sensors (e.g. laser range scanners, stereo cameras, or RGBD sensors) in the form of point clouds. Data from these sensors is first passed through a series of filters. An initial filter attempts to remove spurious data points that can often appear in point cloud data, especially those associated with depth discontinuities in the scene. A second filter uses knowledge of the position of the robot parts to filter out sensor data corresponding to sensor hits on the robot's parts. The resulting point cloud is less noisy than the original data and includes only data corresponding to the environment. A series of snapshots in Fig. 3 illustrate this process.

The environment model uses two representations: a probabilistic occupancy grid representation for unmodeled or unrecognized parts of the environment and a semantic representation for known parts of the environment. The semantic representation is built using a generic, ROS-based interface to object segmentation and detection algorithms. Our current implementation of these algorithms operates on the filtered 3D point clouds to segment support surfaces (such as tables or shelves) and objects on the support surfaces. A database of pre-built 3D models for common household objects is then used to recognize and register some of the objects in the scene. Mesh models of these objects are used to represent them in the environment model. More details of this process are explained in the section on grasping (Section V).

Unmodeled parts of the environment are incorporated into an octree-based representation (called Octomap) [19]. Octomap is an occupancy-grid-like probabilistic representation of the environment that can account for *unknown* space, i.e. space that has not been observed by the robot's sensors. It also maintains a persistent view of the environment, automatically incorporating new sensor data and clearing out obstacles that might have moved away. The environment model serves as the primary source of input for collision checking and is used by both the motion planning and grasping components. Fig. 4 illustrates a typical environment generated using this process.

B. Motion Planning and Execution

1) Moving the Arms: An overall schematic representation of our motion planning and execution framework used for the robot's arms is shown in Fig. 5. The particular implementation we describe here is intended for use with robotic arms. Motion planning and execution for the base is decoupled from that for the arms, and is described in Section IV-B3. Future work will involve the development of components for whole-body

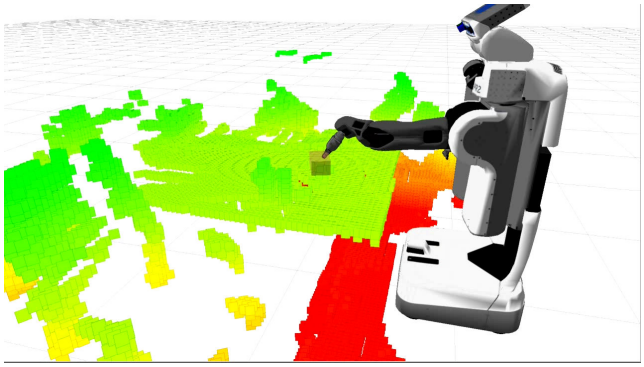

Fig. 4. A typical environment model for a tabletop manipulation task. (This figure is best viewed in color).

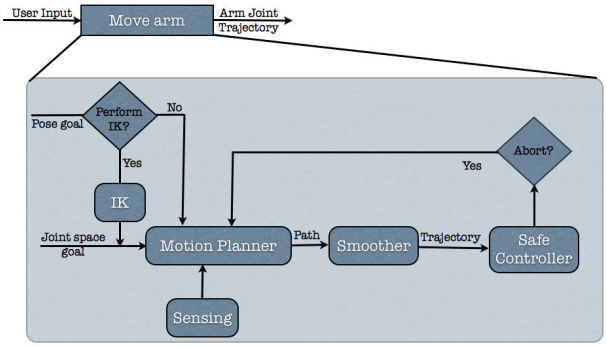

Fig. 5. System architecture for motion planning and execution. *Move Arm* is the software component that serves as the interface to motion planning and execution.

planning and control. Our approach to motion planning and execution follows a traditional *sense-plan-act* paradigm but also incorporates an online monitoring component, allowing our system to abort execution and replan quickly if necessary when new obstacles move into the planned path of the robot.

The **Motion Planner** component in Fig. 5 is carried out using randomized motion planners integrated from the OMPL Library [20]. These motion planners can operate on both joint space or end-effector pose goals (Fig. 5). Inverse kinematics (**IK** component in Fig. 5) is used in the latter case to sample joint-space goals that the planner can then plan to. The output of the motion planner is a path that can often be jagged and long. The path is refined and parametrized in time using *shortcutting* techniques designed for smoothing and shortening the path (the **smoother** component in Fig. 5). Our particular implementation of shortcutting uses cubic splines and can account for bounds on joint velocities and accelerations. A custom analytical solution for inverse kinematics was used for the 7-DOF PR2 robot, parameterized by one of the joint angles. For robots other than the PR2, we use the KDL toolchain [21] to provide a numerical inverse kinematics solver.

Execution of trajectories is handled using a state machine approach that incorporates *active monitoring* of the trajectories executed on the arm. A controller designed to accurately track the desired trajectory is used to execute the planned trajectory on the robot. Active monitoring involves moving the head of the robot (on which the sensors are mounted) to maintain

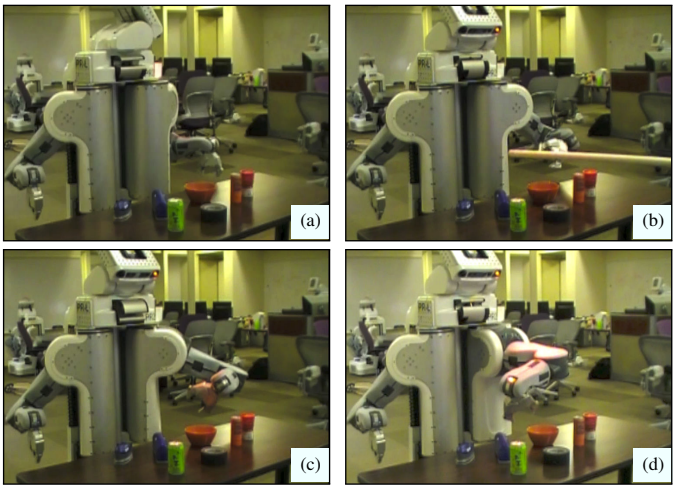

Fig. 6. Active Monitoring: The PR2 starts to execute a trajectory in (a). A new obstacle is introduced into the environment and further trajectory execution is aborted in (b). The PR2 executes the remaining part of the trajectory once the object has cleared away in (c) ending at its final goal in (d).

visibility of the arm as it is executing the trajectory. If a new obstacle is detected along the expected execution path, the arm is brought to a stop along the path. If the obstacle clears out of the path of the robot within a specified timeout, the trajectory continues to be executed. If the timeout is exceeded, a new path is planned and executed, taking into account the presence of the new obstacle. Fig. 6 shows two snapshots from a scenario where the robot stops when it sees a new obstacle, waits for the obstacle to clear, and then re-executes its planned path [22].

A constraint that is often specified in manipulation tasks is to keep an object horizontal, e.g., if it contains liquids. Such constraints can be dealt with during motion planning. For instance, when randomized planners are deployed, one can use projection techniques that attempt to project a random sample onto the constraint manifold [23]. However, we choose to take a simpler approach to such constrained planning by working directly in task space, i.e., in the space of position and orientation of the end-effector of the robot. A randomized planner is configured to function in this 7 dimensional space: $(x, y, z, roll, pitch, yaw, \phi)$. Here (x, y, z) represents the position of the end-effector, $(roll, pitch, yaw)$ represents the orientation of the end-effector in a coordinate frame attached to the base of the robot, and ϕ represents the redundant degree of freedom in the 7-DOF PR2 arm. In this space, an orientation constraint on the gripper can be easily accounted for by placing appropriate bounds on the roll, pitch and yaw states of the end-effector. Inverse kinematics is used to transform these states back into joint space for collision checking. Samples are rejected if no inverse kinematics solutions are found for a particular sample. Fig. 7 shows the result of a constrained plan executed on the robot while carrying out a tabletop manipulation task [22]. Note that this approach can be extended to other 7-DOF arms (but may not work for arms with higher DOF). To the best of the authors' knowledge, this approach to constrained planning for 7 DOF manipulators is novel.

Fig. 7. A constrained motion plan being executed by the PR2 robot while picking up an object and moving it to the side. Note that the object is kept close to horizontal throughout the plan.

Total motion plan calls	274
Successful calls	238
Paths in collision	16
Constraints violated	1
Planning time (mean)	0.904

TABLE I
STATISTICS FOR THE MOTION PLANNING COMPONENT.

2) Evaluation: Table I shows data for multiple runs of the motion planners as part of a pick and place task. The task involved a continuous run of our system using the PR2 robot, where the robot was asked to move objects on a table (without any human intervention). The planner was always given a maximum of 5 seconds, and the smoother was always given 2 seconds for smoothing. Robustness is one of our primary goals; better performance (in terms of speed of the motion plan) could be achieved with more tuning of the motion planners. Failures were mostly the result of noisy data causing the appearance of non-existent obstacles that blocked the motion of the arms. However, the environment representation would often clear out these non-existent obstacle after further sensor sweeps of the workspace. This would then allow the motion planner to successfully plan, and thus the overall system was fairly robust to such failures. Some planned paths would appear to be in collision with the environment before execution, due to either noisy data or people moving into the workspace of the robot. In such cases, the robot would not move, but was able to recover robustly after a short time.

3) 3D navigation: True mobile manipulation requires the ability to navigate through a cluttered environment while carrying objects. In [24], we presented an approach to this problem (which we call *3D navigation*). Our approach combines a representation of large-scale environments using Octomap with a hierarchical collision checking scheme integrated with anytime motion planners. Multi-layered 2D costmaps and a layered decomposition of the robot body are used to reduce the number of full-body 3D collision checks required during planning, thus providing motion plans within reasonable times in cluttered environments. This approach was successfully implemented on the PR2 robot, thus allowing it to navigate with a large object(a laundry basket) in a cluttered environment (Fig. 8). Note that our current implementation of 3D navigation does not account for (or plan for) the arms moving while the base is also in motion. It also does not plan the grasps for large objects like the laundry basket. Future work is intended for developing whole-body grasping capabilities.

V. GRASPING

We define grasping as the ability to secure an object inside the robot's end-effector while resisting external disturbances. It is a key prerequisite for enabling a large number of mobile manipulation tasks, such as object transport and retrieval, tool use, *etc*. Robotic grasping is common in industrial environments, where pick-and-place robots are able to execute grasps with a high degree of reliability. However, this level of performance is usually achieved by taking advantage of the structure inherent in industrial settings: a small and well-defined set of objects to be manipulated, which allows for the design of dedicated end-effectors for known object poses. In environments lacking this kind of structure, such as typical homes or offices, robotic grasping is still an open area of research.

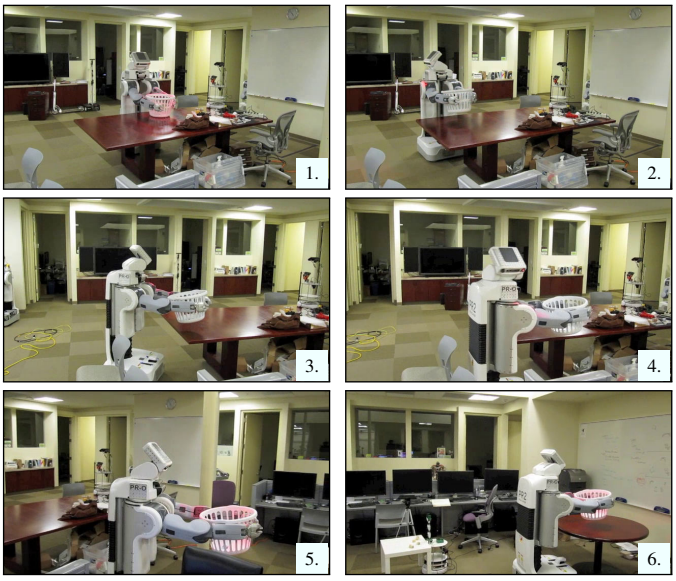

Fig. 8. Sequence of snapshots showing the PR2 robot navigating autonomously with a laundry basket in a cluttered environment.

We believe two of the key problems that must be addressed for robust grasping in unstructured environments are *variability*, requiring the ability to handle a very large set of possible objects and scenarios, and *uncertainty*, requiring the ability to cope with errors ranging from sensing and scene interpretation to mechanism calibration. In this section, we describe our initial work on grasp planning and execution for a wide range of typical household objects, followed by the methods we developed to improve their handling of uncertainty and thus the overall reliability of the system.

A. Grasp Planning and Execution

The ability to plan a grasp for an object depends on the nature and quality of information available to the grasp planner. In environments such as households or offices, the grasp planner receives this information from a perception module that attempts to parse and label the scene. While scene interpretation algorithms are constantly improving, they still can not achieve perfect reliability in general human settings. We believe that a robot must be equipped to both use high-level perception results when possible, and also cope with situations when less information is available.

1) Cluster-based Planner: The first grasp planning algorithm we present uses individual point clouds of graspable objects. It requires that a point cloud from a depth camera,

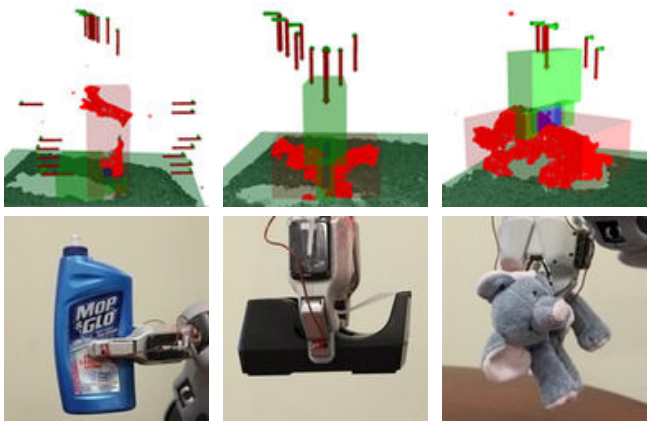

Fig. 9. Cluster-based grasp planning. Top row: individual object point clouds (red points) and grasps (red arrows) associated with them. Bottom row: execution of planned grasps.

such as the Microsoft Kinect® or the PR2 stereo pairs, be separated in individual "clusters", each belonging to one target object. Our basic implementation of the perception module associated with this planner makes a set of assumptions exploiting the fact that household objects are typically found on flat surfaces such as table or counter tops. We compute a planar fit to the point cloud to identify the support surface, and then use Euclidean clustering on the points above it to obtain individual object clusters.

Once an object's point cloud has been segmented, grasps are computed using heuristics based on both the overall shape of the object and its local features. The intuition behind this approach is that many human-designed objects can be grasped by aligning the hand with the object's principal axes, starting from either above or to the side of the object, and searching for parts of the object that fit inside the hand; if no such grasps are found, then grasps from above of high points on the object are also likely to be useful. Grasps found according to these principles are then ranked using a small set of simple feature weights, including the number of sensed object points that fit inside the gripper, distance from object center, *etc*. A number of examples are shown in Fig. 9, and additional information about this component can be found in [25].

A simple extension of this method to deal with cluttered table scenes, or scenes that do not include a flat surface, is to run the same planner on point clusters resulting from a more general segmentation algorithm. One such example of grasping in a cluttered scene, used successfully in [26], is to remove all points whose estimated normals are not nearly-parallel with a chosen grasp approach direction (for instance, grasps from above tend to be useful for cluttered tables, and grasps from the front tend to be useful for shelves), and to then run Euclidean clustering on the remaining points. Planning grasps for the resulting clusters generates useful, axis-aligned grasps of the tops (or front faces) of objects.

2) Recognition-based Database Planner: If an object recognition component is present as part of the robot's perception toolset, additional information regarding a recognized model can be pulled from a database of known objects. Complementing the methods and algorithms discussed here, we have introduced a database of common household objects available from major U.S. retailers. For each object, the database contains a triangular surface mesh, as well as additional tags such as maker and model, barcode, class labels, *etc*. This dataset, described in detail in [27], is available as a relational database using the SQL standard.

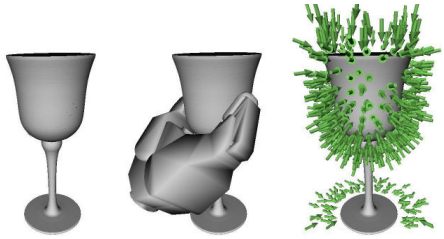

Fig. 10. Grasp planning in a simulated environment for a known object (the mesh, a simulated grasp and the complete set of pre-computed grasps).

When the target is a recognized database object, the robot can plan grasps using the complete object mesh. Furthermore, grasps for each object can be pre-computed in advanced and also stored in the database. For each object in the database, we used the *GraspIt!* simulator [28] to pre-compute a large number of grasp points for the PR2 gripper. Using the grasp quality function and search method described in [27], we performed off-line grasp planning for the objects in our database, allowing 4 hours of planning time per object. This process resulted in an average of 600 pre-computed grasp points saved in the database for each object, as examplified in Fig. 10.

We use the database of known objects and grasps in conjunction with an object recognition module that attempts to match each object point cluster, segmented as described in the previous section, to each of the object meshes in a pre-defined set. Matching is performed using an iterative technique similar to the ICP algorithm [29], and the best fit is returned. Other recent recognition methods, such as the MOPED framework [30], have shown the ability to recognize objects even when stacked closely together, and could be used to allow the robot to perform grasps in highly cluttered scenes.

3) Grasp Execution: The grasps generated by either of the two planners discussed above consist of end-effector poses relative to the target. After planning, a collision-aware Inverse Kinematics module checks each grasp for feasibility in the current collision environment, constructed as described in the previous section. Once a grasp is deemed feasible, the motion planner generates an arm trajectory for achieving the grasp position (as described in Section IV), and the grasp is executed.

B. Coping with Uncertainty

The methods described so far can be used to grasp a wide range of common household objects, taking advantage of high-level information (such as object identity) when it is available but also handling novel objects when needed. In practice, however, we also found them to be sensitive to errors in either the perception or execution modules, where an object was misrecognized, segmentation was incorrect, or calibration between the sensors and the end-effector was imperfect.

1) Probabilistic Grasp Planning: One method of dealing with uncertainty in object shape or pose due to such errors

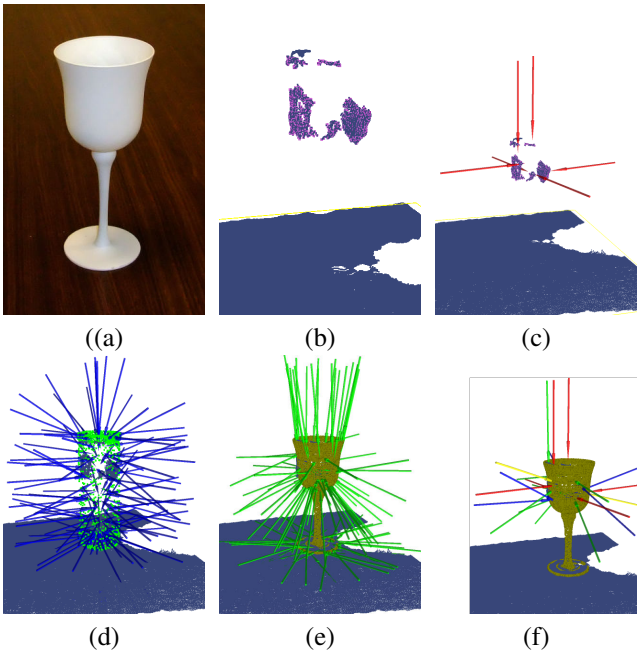

Fig. 11. Combining multiple grasp planners. **(a)** A cup to be grasped, and the point cloud as seen by the stereo camera **(b)**. **(c)-(e)** Different object representations, and grasps planned for each. These include grasps planned on the segmented point cluster (c), and on meshes for results from object recognizers, one incorrect shown in (d) and one correct shown in (e). **(f)** A set of grasps that combines information from all of them.

is to combine information from multiple object recognizers, grasp planners, and/or grasp evaluators. Fig. 11 illustrates the general concept: when trying to grasp the cup shown in (a), the robot's stereo cameras see only the point cloud shown in (b). The robot's object recognizers provide two hypotheses, with corresponding quality scores: the tennis ball can (incorrect) shown in (d) and the cup (correct) shown in (e). Grasps can be planned on all possible hypotheses, including different object shapes and poses (using, for instance, GraspIt!), and even just the segmented point cluster (using, for instance, the cluster grasp planner described in section V-A).

The entire pool of grasps, generated on all available hypotheses, can be evaluated using any number of available grasp evaluators. In this case, the grasp evaluators used are the same ones used to generate the grasps: *GraspIt!* can be used to evaluate arbitrary grasps on each possible object mesh/pose, and the cluster grasp planner can be used to evaluate arbitrary grasps based just on the segmented cluster. By combining both object recognition results and grasp evaluation results, we can estimate the probability of success for each grasp in the generated pool, and thus select only those grasps most likely to succeed. In one experiment that compared probabilistic and threshold-based versions of our standard grasp pipeline while grasping 25 different objects, the grasp success rate increased from 18 out of 25 (72%) to 22 out of 25 (88%) for novel objects, and from 21 out of 25 (84%) to 22 out of 25 (88%) on database objects. More details are available in [31].

2) Reactive Grasp Execution: Grasping objects based on visual information can be affected by errors in object localization, perceived shape, or calibration between the robot's cameras and its end-effectors. We can compensate for such errors by using tactile information acquired while grasping. The PR2 gripper is equipped with fingertip capacitive sensor arrays, each consisting of 22 individual cells distributed on the tips, sides and inner pads. During the final stages of the grasp, we use a set of "reactive behaviors" based on information from these sensors to adapt the grasp to unexpected contacts.

The first reactive behavior attempts to maneuver the gripper around the object when unexpected contacts are detected by the fingertips during the approach, by backing up and moving in the direction of the contact. The second behavior executes a *compliant grasp* when one fingertip contacts the object before the other: the arm is moved in coordination with the gripper to keep that fingertip in place, preventing the object from being pushed out of the grasp while the gripper closes. The final behavior adjusts grasps that are likely to be unstable, as judged by seeing contacts only at the fingertips or only on one side of the fingertip sensor arrays, by shifting the end-effector in an attempt to center the contact points between the fingertips. Although these behaviors are simple and deal with objects in a model-free way, they are successful in fixing many minor errors that would have caused the grasp to fail. In one experiment involving 68 grasps of 30 different objects, where open-loop grasping using the cluster-based grasp planner successfully grasped the object in 60 out of 68 attempts (88%), adding reactive grasp execution increased the success rate to 66 out of 68 (97%). More details on the reactive grasping behaviors can be found in [25].

3) Adaptive Grasp Force Control: A further issue when grasping and transporting objects is the need to regulate the internal forces applied to the object: too much force, and deformable objects will be crushed and potentially damaged; too little force, and objects will be dropped. A control scheme that uses the PR2's tactile sensors and accelerometers for regulating grasp forces is presented in [32]. Real-time data from the gripper's fingertip tactile arrays and accelerometers are combined into signals designed to mimic three different human skin sensors (SA-I, FA-I, and FA-II). These signals are then used in event-based, force-regulating controllers that enable the PR2 to grasp soft objects without crushing them, to grasp harder when an object is slipping in the hand, and to release objects while placing when the object hits the table.

VI. Portability to Other Manipulation Platforms

Our algorithms and implementations for collision-free arm motion, grasping, and manipulation are currently in use on PR2s all around the world; however, to maximize the impact of our work we have also sought to make it straightforward to use our work on any manipulator. The vision of this work is to allow a user to take the physical description of his/her robot – in URDF or COLLADA format – and quickly and interactively configure a system that will allow them to manipulate objects in either a simulated or physical environment.

For the first stage of this work we focused on allowing new users to configure our software for generating collision-free arm trajectories interactively using a tool called the Arm Navigation Wizard. This tool requires the physical specification of the robot and some user input; it produces all necessary configuration and application files for collision-free

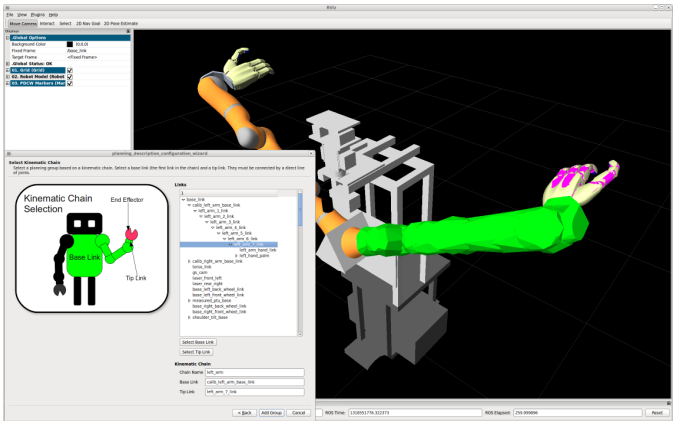

Fig. 12. The Arm Navigation Wizard window being used to set up a kinematic chain group for the TUM-Rosie robot.

arm navigation, including those for self- and environment collision checking, joint- and task-space planning, and inverse kinematics.

The physical specification of the robot is assumed to contain strictly mechanical properties of the robot: joint locations and limits, and link locations and geometry. It lacks some first-order semantic information about which joints and links constitute the robot's arm or arms, and end effector or end effectors. We require that users of the Arm Navigation Wizard configure at least one group that can be used for planning. For most users this group will consist of a kinematic chain corresponding to a robot arm. The tool makes it easy to navigate a robot's kinematic tree to determine the base and tip links of the kinematic chain, and upon creation the chain is rendered graphically. Configuring groups for a robot's arm or arms is the only required interaction in the Wizard.

The primary difficulty in configuring our motion planning and execution system for a new robot involves configuring the self-collision checking capabilities. Our software automatically catagorizes pairs of links in a robot into 3 categories: (1) links that never collide with each other, (2) links that always collide with each other or (3) links that may sometimes collide with each other. A uniform joint space sampling is used to generate multiple configurations that are tested exhaustively for collisions between different pairs of links. Data from these tests is used to determine which category every pair of links falls into. This information is cached and reused during motion planning to significantly speed up collision checking. We note that precisely determining which link pairs will never be in collision is a difficult problem; future work will involve dynamically focusing the samples on areas of the joint space that produce more information.

The Arm Navigation Wizard allows users to get our system working on their robots in minutes or hours, whereas it previously may have taken months. With this capability, and working in collaboration with two researchers from Fraunhofer IPA, we were able to port the manipulation capabilities described in Sections IV and V to the Care-O-Bot [8]. The process, from generating configuration files using the Wizard to executing trajectories on the Care-O-Bot, took only a few hours. Porting our grasping and manipulation capabilities took more time (a few days), but the end result was that both robots could essentially run the same code despite having different sensing hardware, arms, and end-effectors, as shown in Fig. 1. Future work involves developing cross-robot benchmarking suites for motion planning and extending the Wizard to grasping and manipulation.

VII. Applications

The system presented in this paper has been used in several applications, developed both by our group and by external groups. All the code developed for this system is available in the ROS software platform and can be easily downloaded and installed on new robots(www.ros.org/wiki). The arm navigation framework includes complete instructions [33] on installation and running the software for application on a new robot. The first application built up using our system was an autonomous pick and place demo for tabletop manipulation. The application is designed to be completely autonomous and has a simple goal: to move objects around on planar support surfaces, e.g. tables or counters. Two independent actions form the basis of the application: a pick action to pick up objects and a place action to place objects on a support surface. The pick and place application uses these two actions to move all objects on one side of a table to the opposite side, then switches sides and continues. Fig. 13(a) shows an example of this application in action. As mentioned earlier, the system works both with known objects, for which grasps have been pre-computed, and also with novel objects, for which new grasps need to be constructed.

In [34], an autonomous beverage fetching application was designed, based around the motion planning components of our system. The application used a state-machine-like architecture for automated retreival of beverages from a fridge based on user requests. It also imposed constraints on the motion plans to keep the beverage containers upright while they are being manipulated. Fig. 13(b) shows a snapshot from the application in action.

Our system was further combined with a cart-pushing application to execute a cleanup application. The cart allowed the PR2 to carry multiple objects on a cluttered tabletop through a cluttered office environment. The PR2 loaded the cart autonomously using our system, although the choice of objects to pick up and retrieve was made by a human. Fig. 13(c) shows a snapshot from this application.

Our system has further been used by several external groups. In [35], our system was used in the PR2 Remote Lab. The lab was intended to allow remote participants to use a PR2 robot. A web interface that uses our systems allowed remote users to pick and place objects on a tabletop, and the application was successful in allowing users to manipulate a wide variety of objects. Our system also served as a testbed for several other researchers integrating new motion planning algorithms [16], [17], and is also used in a cookie baking application in [36], where the PR2 robot manipulated several items in a kitchen tabletop environment.

Fig. 13. Applications of our system (from left to right): (a) A pick and place application for moving objects. (b) A beverage retrieval application. (c) The PR2 using a cart to carry objects in a cluttered office environment.

VIII. Lessons Learned

Our experience with real-world deployment of the mobile pick and place system has clearly showed the importance of having good situational awareness in human environments. In particular, we have found a probabilistic representation to be important, especially in dealing with noisy sensors and moving obstacles in the environment (especially humans). The separation in the representation of objects from the rest of the environment has been very helpful - allowing for the maintanance of object models at the higher resolution required for manipulation (e.g. grasping) while allowing other parts of the environment to be maintained at a less expensive lower resolution for motion planning. We have also found that semantic representations of relevant parts of the environment, e.g. tables or shelves, aids in high-level decision-making by abstracting information away from a detailed, grid-like representation. Finally, maintaining a persistent model of the environment is important for global planning, while a faster, reactive model of the environment can be used for more reactive local planning or for the checking of trajectories.

In practice, we have found that the quality of motion plans is a significant factor in people's perception of the robot's actions. Thus, post-processing (or smoothing) of paths generated by the randomized planners is essential before execution. Imposition of velocity and acceleration constraints also significantly helped in providing easier-to-follow trajectories for the controllers. In grasping tasks, traditional motion planning, i.e. planning for paths that are not in contact with obstacles, was best suited for planning to a *pre-grasp* position offset from the actual grasp position. A reactive approach using tactile sensing was ideal for the last part of the grasp, i.e. for any actions where the arm was moving into and out of contact, allowing the robot to account for uncertainty in the object's pose or occluded obstacles not visible to the visual sensors. Active monitoring of path execution was very useful in situations where humans were present in proximity to the robot, especially when they were moving in and out of the robot's workspace. Significant parts of the workspace, however, are not visible to the robot's visual sensors. Whole body sensing (e.g. robot skin) or other types of collision and contact detection will thus be particularly important for the successful deployment of mobile manipulation systems in human environments.

IX. Conclusions

In this paper, we have presented an integrated approach to mobile pick and place tasks in unstructured environments. Our approach has been successfully used in multiple applications and across robotic platforms, demonstrating its versatility and applicability in varied environments. It has also served as a testbed for integrating new approaches to motion planning, grasping and perception. We intend to explore more applications and extend our approach to other domains in order to develop a reliable and robust mobile manipulation system that can be applied to a wide variety of tasks.

We have also made an attempt to make our approach easy to integrate on other mobile manipulation systems. We intend to continue building tools that would make configuration of more components of our system easier. The use of our system on multiple platforms has helped us to continuously improve and update our systems, and has also exposed shortcomings in our system, which we aim to address through the development of new techniques and approaches. While a certain amount of robot-specific configuration is inevitable in mobile manipulation, we believe that the core components and capabilities that we are developing can be easily used across a wide variety of robots.

There are several areas where there is ample scope for improvement. We would like to achieve better integration with proprioceptive sensors and force or torque information now available on various force-controlled robots. We would also like to extend the capabilities of robots such as the PR2 by enabling dual-arm or whole-body manipulation, thus allowing such robots to manipulate much larger objects. Furthermore, we aim to take advantage of the probabilistic information available in Octomap, using it to better account for uncertainty in the environment. Finally, we would like to plan for and execute more coordinated motions of the base and arms of a robot while using the sensors to maintain better coverage of the environment.

X. Acknowledgements

We gratefully acknowledge the contributions of the entire PR2 and ROS development teams at Willow Garage and the members of the PR2 Beta Community. We also acknowledge in particular several interns and collaborators

who have contributed to the development of the manipulation stacks in ROS, including Ioan Sucan, Mrinal Kalakrishnan, Ben Cohen, Michael Phillips, Armin Hornung, Kai Wurm, David Lu!!, Rosen Diankov, Peter Brook, Adam Leeper, Joe Romano, Katherine Kuchenbecker, Gunter Niemeyer, Mark Moll, Maxim Likhachev, and Lydia Kavraki.

REFERENCES

[1] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Sucan, "Towards reliable grasping and manipulation in household environments," in *ISER*, New Delhi, India, December 2010.

[2] S. Srinivasa, D. Ferguson, M. V. Weghe, R. Diankov, D. Berenson, C. Helfrich, and H. Strasdat, "The Robotic Busboy: Steps Towards Developing a Mobile Robotic Home Assistant," in *Intl. Conference on Intelligent Autonomous Systems (IAS-10)*, July 2008.

[3] M. Dogar and S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, October 2010.

[4] B. Bauml, T. Wimbock, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *International Conference on Intelligent Robots and Systems*, 18–22 Oct 2010, pp. 2592–2599.

[5] N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann, "Integrated grasp and motion planning," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 2883–2888.

[6] A. Jain and C. Kemp, "EL-E: an assistive mobile manipulator that autonomously fetches objects from flat surfaces," *Autonomous Robots*, 2010.

[7] A. Saxena, L. Wong, and A. Ng, "Learning grasp strategies with partial shape information," in *AAAI Conf. on Artificial Intelligence*, 2008.

[8] C. Partlitz, M. Hagele, P. Klein, J. Seifert, and K. Dautenhahn, "Care-o-bot 3 - rationale for human-robot interaction design," in *Intl. Symposium on Robotics*, 2008.

[9] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth, "Robotic roommates making pancakes," in *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.

[10] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *IEEE Intl. Conf. on Robotics and Automation*, 2010.

[11] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. VandeWeghe, "Herb: A home exploring robotic butler," 2009, doi:10.1007/s10514-009-9160-9.

[12] S. Edwards, "The ros industrial platform," http://ros.org/wiki/industrial, 2012.

[13] "Adept technology applies open source ros software to industry," http://ir.adept.com/releasedetail.cfm?ReleaseID=648907, 2012.

[14] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[15] I. A. Sucan, M. Kalakrishnan, and S. Chitta, "Combining planning techniques for manipulation using realtime perception," in *International Conference on Robotics and Automation*, 2010, pp. 2895–2901.

[16] B. Cohen, G. Subramanian, S. Chitta, and M. Likhachev, "Planning for manipulation with adaptive motion primitives," in *ICRA*, Shanghai, China, 2011.

[17] P. Vernaza and D. D. Lee, "Learning dimensional descent planning for a highly-articulated robot arm," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2011. [Online]. Available: http://www.seas.upenn.edu/~vernaza/papers/lddPR2.pdf

[18] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[19] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010, software available at http://octomap.sf.net/. [Online]. Available: http://octomap.sf.net/

[20] "The Open Motion Planning Library (OMPL)," http://ompl.kavrakilab.org, 2010.

[21] "The Kinematics Dynamics Library (KDL)," http://www.orocos.org/kdl.

[22] "Constrained planning and active monitoring on the pr2 robot," http://vault.willowgarage.com/wgdata1/vol1/ram_2011_mobile_manipulation/planning_and_execution.m4v, 2011.

[23] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *IEEE International Conference on Robotics and Automation*, May 2009.

[24] A. Hornung, E. Jones, S. Chitta, M. Bennewitz, M. Phillips, and M. Likhachev, "Towards navigation in three-dimensional cluttered environments," in *The PR2 Workshop: Results, Challenges and Lessons Learned in Advancing Robots with a Common Platform, IROS*, 2011. [Online]. Available: http://vault.willowgarage.com/wgdata1/vol1/iros_2011_pr2_workshop/hornung_freiburg.pdf

[25] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, "Contact-reactive grasping of objects with partial shape information," in *IROS*, 10 2010.

[26] A. Leeper, K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow, "Strategies for human-in-the-loop robotic grasping," in *HRI*, 2011.

[27] M. Ciocarlie, C. Pantofaru, K. Hsiao, G. Bradski, P. Brook, and E. Dreyfuss, "A side of data with my robot: Three datasets for mobile manipulation in human environments," *IEEE Rob. and Autom. Mag.*, vol. 18, no. 2, 2011.

[28] A. Miller and P. K. Allen, "GraspIt!: A Versatile Simulator for Robotic Grasping," *IEEE Rob. and Autom. Mag.*, vol. 11, no. 4, 2004.

[29] P. J. Besl and M. I. Warren, "A Method for Registration of 3-D Shapes," *IEEE Trans. on Pattern Analysis*, vol. 14, no. 2, pp. 239–256, 1992.

[30] A. Collet, M. Martinez, and S. S. Srinivasa, "The MOPED framework: Object Recognition and Pose Estimation for Manipulation," 2011.

[31] K. Hsiao, M. Ciocarlie, and P. Brook, "Bayesian grasp planning," in *ICRA 2011 Workshop on Mobile Manipulation: Integrating Perception and Manipulation*, 2011.

[32] J. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, "Human-inspired robotic grasp control with tactile sensing," *IEEE Transactions on Robotics*, In Press.

[33] S. Chitta, G. E. Jones, I. Sucan, M. Moll, and L. Kavraki, "Iros tutorial on motion planning for real robots," http://www.kavrakilab.org/sites/default/files/iros2011_ros_ompl_tutorial.pdf, 2011.

[34] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mosenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the pr2," in *ICRA*, Shanghai, China, 05/2011 2011.

[35] B. Pitzer, S. Osentoski, P. Roan, C. Bersh, and J. Becker, "Making robots cheaper, more capable, and safer," in *The PR2 Workshop: Results, Challenges and Lessons Learned in Advancing Robots with a Common Platform, IROS*, 2011.

[36] M. Bollini, J. Barry, and D. Rus, "Bakebot: Baking cookies with the pr2," in *The PR2 Workshop: Results, Challenges and Lessons Learned in Advancing Robots with a Common Platform, IROS*, 2011.